The 68xxx Machines

Vol. 1, Issue 6, July 1991

Price: $2.75

The 68xxx Machines is published and copyright (C) 1991 by Chatham House Company, RD#1 Box 375, Wyoming, DE 19934. Ph. (302) 492-8511. The editor is James H. DeStafeno. One year USA subscription is $12.50. Canada and Mexico $14.25. All others (surface) $17.25. All major credit cards accepted. Our low prices reflect a 10% cash discount. Please add 10% to credit card orders. Any size display advertising is accepted. The half page rate is $10/issue. Write for other size/duration rates.

Readers are encouraged to contribute letters, articles, programming information and other material related to computers with the 68xx(x) processors; excepting Macs and Amigas. Please send material to the above address. Thank you for your support.

This Issue:

Editor's Thoughts

by Jim DeStafeno

Another mile stone has been passed. All the people involved with *The 68xxx Machines* deserve a pat on the back. This is the sixth month, half year, issue. "Thank you" to those that have voted their encouragement with articles. By now you must have noted they are not a rehash, but rather are current and contain information that can be used, today.

"Thank you" to those that are advertising. I have not heard one complaint about the products or services. And of course "Thank you" to the subscribers. Without you all it would be a pretty useless exercise.

To try to obtain better results, we're experimenting with different character fonts and with a new method of producing the copy that eventually goes to our printer. The new font allows us to increase the amount of material we can publish without reducing the size of the type. Hopefully the new font will also be easier for you to read.

Speaking of our printer, Randy Krippner, who does some of the desk top publishing work to put *68xxx* together, omitted the two year price of a subscription for foreign countries on the subscription coupon. When I mentioned it to Mike, our man at the printer, he said he could "pencil" it in. Did you notice what he did? Mike, thank you for the effort, but you didn't fool any one. No one sent $133.50; guess we better go back to the intended $33.50.

Got a letter from Paul Ward. You know, the main MM/1 man. He very nicely said some of my observations printed in the May '91 issue of his machine at the Chicago RAINBOWfest, were out of order. He said that: "The MM/1 was showing animations and graphics in fine color. They were not dim in any way. In fact, the MM/1 display is as bright or brighter than *either of our competitors." (The italics are mine, ED.)* Okay, sorry if what I saw was not what I thought it was; maybe I was looking at night scene in an adventure game.

In another part of the letter, Paul commented about my remarks relating to the moderated 'talk' between Ed Gresick, DELMAR; Frank Hogg, FHL and himself, "Frank continually gave me a hard time because our booth sign said the MM/1 was *faster then competing systems. (The italics are mine, ED.)* Anyone who has seen our brochure realizes that we compare ourselves prominently to systems that we do beat in speed." Ahaa Paul, not sure competitore can mean particular companies in one reference, but not those same companies in the next.

There are other points in the letter such as IMS has been delivering computers longer then DELMAR. Humm, maybe so Paul, but will you send me a name of a purchaser that has received a complete MM/1 with a working hard disk; I'd like to print a taped telephone conversation review for our readers.

On another day Frank Hogg called. We kicked around a couple of things, like my comments on the "fest". For one, he says he doesn't like an inference that his machine is slow; especially when his and the MM/1 use the same CPU and clock speed. He also gave a little insight on cost. Both his and the DELMAR machines are shipped with their expansion buses in place, while the MM/1's is not. At the 'talk' Paul said the MM/1's will be an upgrade. Frank allowed, an expansion add-on bus will not be cheap, and expects the final cost of each machine will be about the same, if the MM/1's isn't more than the others.

Addressing my question about lead time, Frank said the TC-70 lead time is running 4 to 6 weeks due to the lead time on some parts.

He also said he would send a press release sheet on two new products. One a 16Mb Ram Disk board for the K-Bus and second, a new 68030 based computer with lots of serial ports. The hard copy hasn't arrived as yet.

What else? Not much; Gresick is quiet. Too busy shipping computers I guess. Be talking with you next month, take care for now, and thanks again to everyone for the six months of support. JHD

CLASSIFIED ADS

SALE: PT68K-2 complete system: 10MHz motherbd w/1M RAM, 20M hard drive, 720K floppy, amber monitor, SK*DOS, RBASIC, EDDI, all manuals, more! Used very little, $800 + shipping. Phil Krieg, 11221/2 Bayland, Houston TX 77009. (713) 861-0367 eve/wkends.

SALE: 4 each TM848-2 Tandon 8" HH Drives, brand new, with service manual, $249 each. 1 each SS50 /SS30 Smoke Signal 6809 System, 2 Mhz, 48K RAM, Mother Board, FDC 5"/3" 80 tracks, UIA, ACIA, PIA, Extender Boards SS50/SS30, male and female connectors (gold), others. Best Offer. Shipping Included, all items. Geza Holzhaker, Apdo. Correos # 393, Merida 5101, Venezuela, FAX: 58-74 63.5857 (nights, East time.)

"WANTED", $2.50/50 space line/issue.
"SALE", $5.00/50 space line/issue.

OSK And Super Terminals

by Bob van der Poel

Working with OSK, a system designed to be adaptable to different types of peripherals, including different terminals can be fun. Such flexibility often results in challenging problems in search of solutions. Things which are trivial on a "fixed" system like MS-DOS, become the subject of endless debate and excuses to stay up late in the computer room. When designing my VED text editor I was faced with such a "problem."

The VED uses the OSK 'C' Termcap Library to handle the terminal output. The Termcap Library is a set of C routines which read a data file containing the information needed to do terminal output, attributes like highlight, underline and blinking; plus the size of the screen, cursor positioning, etc. To extract the correct information from the data file the SHELL variable "TERM" is set to the name of the terminal being used. All this works smoothly, once the termcap data file is set up properly; except for one small detail, the screen size.

The screen size listed in the termcap data base for a particular terminal is, like all the other values for that terminal, a constant. If your terminal is a "White Bread Whiz-Bang" with 80 columns and 22 lines there is no problem. But what if the terminal has variable sized screens? Or if you have a system using a windowing system?

One solution is to have the user enter the screen size as a command line option which will override the termcap values. It works, but it's not too friendly. Another solution, assuming a console system, would be to have the program interrogate the system with a GetStt. Nice idea, but it fails with a remote terminal and assumes that the program knows how to do the interrogation; (Microware has not supplied a standard GetStt code for this).

Another solution might be to have the terminal size set up as a SHELL variable. T this is a good idea, but it creates problems when a windowing system is changing the size often.

The solution I came up with is to use a extension module to let the editor know about the screen size. If the module exists, VED gets the size from it; otherwise it uses the termcap entry. The extension module simply prints the number of rows and columns to standard output. It can be written in any language and can get the size in any manner it wishes, via a GetStt, from some kind of information file or from a SHELL variable. VED has no need to know how the size was determined, it just needs to know what it is.

A simple example of the extension module (called "ved_ssize") would be:

```
main()
{
   printf("%d\n%d\n",80,26);
}
```

Of course, if the need for the program exists a more complex program would be needed (the example would be better handled with the termcap entry).

The only thing remaining is for VED to get the data. The method I use is for VED to call ved_ssize with a pipe opened so VED can read the output from the called program. The following code shows how VED does it:

```
save_std_out=dup(1);/* save stdin */
close(1); /* force pipe to */
open("/pipe",S_IREAD+S_IWRITE);
                        /* use path 1 */
t=os9fork("ved_ssize",0,0,0,0,0,0);
                        /* run sizer */
pipe_path=dup(1);/* save forked prog
                       output */
close(1);
dup(seve_std_out);/*restores stdout*/
close(save_std_out);
if(t>0){         /* fork successful */
 cread(pipe_path,buf2,sizeof(buf2)-1);
                       /* get columns */
   get_exp(&j,buf2,&e);
   if(!e){ /*no error evaluating
                 xsize*/
 cread(pipe_path,buf2,sizeof(buf2)-1);
                          /* get rows */
      get_exp(&i,buf2,&e);
      if(!e) set_ssize(j,i);
  /* go set size */
    }
}
close(pipe_path); /*close pipe peth*/
```

In the above fragment cread() reads a string and appends a null to it, get_exp() evaluates a numeric expression and set_ssize() sets the screen size.

This might not be the most elegant solution, but it does have a number of advantages. First, the code for both the main program and for the auxiliary program is straight forward and easy to write as well as maintain. Second, the main program does not need to make any assumptions about the user's terminal.

In the future I hope to cover more picky problems like this. If you have a request, question or comment please contact me at:

PO Box 57	PO Box 355
Wynndel, BC	Porthill, ID
Canada V0B 2N0	USA 85853

Rush Caley, LIVE!

For years, I could never understand why certain people engaged in activities I deemed "crazy". Here's a few examples of what I mean. Bunji Cord jumping, mountain climbing, rock climbing, white-water rafting, hang-gliding, sky-diving, and many more. I never could dredge up more than a modicum of sympathy when I read headlines about fatal accidents involving this type of activity. But now as my 45th birthday approaches, I've finally reached an understanding of these endeavors. It has nothing to do with bravery, death-wish, or a neurotic need for attention. I now see that these people are merely reacting to the "predicament" of life in one of the two equally moronic alternatives available.

For this is the predicament of life: it's dangerous! I recently heard about the results of a study that showed left handed people die 9 years earlier than their right handed counterparts. The dangers appear when you are very young and continue to multiply and haunt you through life. If you run with the scissors or play with BB guns, you'll put your eye out. If you keep making that face, your mouth will stay that way. If you continue to pick at that, it'll become infected. Loud music will deafen you; and in addition to sitting too close to the television, we all know what will blind us for sure.

Eating too much will make you sick; not eating enough will make you sick; eating the wrong foods will make you sick. Driving too fast is dangerous, driving too slow is dangerous. Too much sun is bad for you, too little sun is bad for you. We live in dire fear of atom bombs, earthquakes, tornadoes, volcanic eruptions, holes in the ozone layer, AIDS, Cancer, and a host of other diseases frightening enough to keep you awake all night.

And if that's not enough, not only are we physically in mortal danger, we are also spiritually at risk. If you have no religion, you'll go to hell. If you have religion, but it's the wrong one, you'll go to hell. Heaven will be a small golden city for those clever enough to get it right.

So how does one cope with all of this disaster that envelops our existence? Well, it appears to me now that there are basically two alternatives to just "plodding ahead" with our lives. The first alternative I mentioned at the outset. It now makes sense to me why people engage in some

*** New For OS-9/68000 ***

Our famous VED text editor is now available for computers running the OS-9/68000 operating system. It uses TERMCAP functions to make it compatible with most terminals. Nearly all of VED's internal settings (macro definitions, key-bindings, editing modes, etc.) can be modified from an initialization file. VED's editing mode options include insert/overstrike modes, automatic indenting and numbering, wordwrap on/off, etc. Standard functions like search, find/replace, blockmove/copy/delete, word and line delete are completely supported with countless variants and options. A complete set of "undo" functions make it easy to correct mistakes. VED also has a built-in text formatter; included in the command set are margin settings, headers and footers, justification modes, and embedded commands for various printer fonts (italic, underline, etc). An interactive preview mode makes it easy to see what the final document will look like when printed. User extendable help screens speed things up when you just need a refresher, rather than using our detailed, indexed 68 page reference guide.

To order VED please send us your check or money order (sorry, no credit cards) for $39.95 plus $3.00 S/H.

Bob van der Poel Software

PO Box 57		PO Box 355
Wynndel, B.C.	or	Porthill, ID
Canada V0B 2N0		USA 83853-0355

Telephone 604-866-5772

of the more intrepid endeavors. It's a retaliatory measure. In other words, a person says: "I'm not going to wait for Cancer, an earthquake, or an out of control Bus. If I've got to go, I'm going to go in a blaze of glory." That's understandable now. It beats the hell out of a hospital bed and tubes jammed into every conceivable opening.

But me? I'm not on my way out to buy any scuba gear just yet. It's not that I'm "chicken"; but even though I have more understanding of those types of risky entertainment, I'm one that just doesn't have the where-with-all to go asking for it. I choose to just "plod ahead" even though it's a jungle out there.

What's the second alternative I alluded to earlier? Not very attractive, I'm afraid. Not really worth the effort to mention it. Just stay in bed and never come out from under the covers. Never go anywhere. Never *do* anything. Play it safe. Some life. All fear, no fun, and wind up just as dead. Like I said, moronic. Well, I'm on my way out the door to the movies. I'll probably even put extra butter and salt on my popcorn.

What Is Minix?

by Joseph Fosco

Several years ago after building a Peripheral Technology PT68K-2 computer kit I tackled getting an Operating System for it. I liked the idea of Unix, but after a little research I found, if available, it would be very expensive.

At first, the closest thing available for the PT68K-2 I could find was OS-9. It looked good, however, it was rather expensive and its source code wasn't available. Not long later I heard about a Unix look-alike called MINIX. I was able to test run it on an IBM PC. Soon I found myself considering doing a port of the system to my PT, (what better way to learn about operating systems?); but as it turned out it was not necessary. Someone had already done it. Before I begin with "Getting MINIX to Run", perhaps a little background on MINIX would be appropriate. Much of the following information comes from the MINIX Information sheet available on USENET.

MINIX is an operating system very similar to UNIX. It was written by

GRANITE COMPUTER SYSTEMS

MODEMS

NEW 9600 Baud V.32/V.42/V.42bis data modems. MNP1-5LAPM.
Error Correction and data compression (much higher effective throughput — as much as 38400 Baud). External — call for price

NEW PRICE 2400 Baud V.42/V.42bis data modems. MNP1-5/LAPM.
Error correction and data compression (much higher effective throughput — as much as 9600 Baud). External $189 (+$7 S&H).

NEW PRODUCT 9600 Baud Send/Receive Fax modems.
Send/Receive text/graphics files from/to your computer/any Fax machine in the world. Full 2400 Baud data modem capabilitiy.
Includes software. External $159/Internal $149 (+$5 S&H)

NEW PRICE 9600 Baud SendFax modems.
Send text/graphics files from your computer to any Fax machine in the world. Full 2400 Baud data modem capability. Includes software. External $129/Internal $119 (+$5 S&H)

NEW PRICE 2400 Baud Data modems External $109/Internal $99 (+$5 S&H)

These are all high quality modems made by Zoom Telephonics in the USA. Fully Hayes compatible. Two year mfg warranty. Software available.

S&H Canada (Air PP and Ins) V.42/V.42bis $12.00 Sendfax/Data $8.00

GCS FILE TRANSFER UTILITIES — Version 3.0

The GCS File Transfer Utilities provide a simple and quick method to transfer text/binary files from/to a variety of floppy disk formats.

Commands PC, RS, FLEX disks: Dir, Dump, Read, Write
PC disks: Rename, Delete, Format

Handles most 5.25 and 3.5 formats. Any level sub-directories (PC). Binary files. Use pipes for multiple file transfers. Multi-Vue version can be used under Multi-Vue or as stand alone Shell commands.

Requires OS-9 L2 for COCO 3, L1 for COCO 1 or 2. 2 drives (one can be hard/ramdisk, one floppy 40 T DD DS). Multi-Vue for Multi-Vue version. SDISK3 for COCO3 - SDISK for COCO 1 or 2. Multi-Vue version $54.95 Standard version $44.95
V3.0 updates (provide disk number) $15.00

D.P. Johnson Software SDISK or SDISK3 $29.95 L1+L2 Utils $75.00
Shipping and handlling — any software $2.00

Orders must be prepaid or COD. VISA/MC accepted. COD is additional.

571 Center Road, Hillsboro, NH 03244 USA
(603) 464-3850

OS-9 is a trademark of Microware Systems Corporation and Motorola, Inc.
MS-DOS is a trademark of Microsoft Corp. FLEX is a trademark of TSC, Inc.

Dr. Andrew S. Tanenbaum for teaching operating system design. MINIX was written from scratch and does not contain any AT&T code. For this reason, source code is made available. The current version of MINIX is 1.5.

MINIX is a full multiprogramming operating system (allows more than one program to run at once). In addition, it is system call compatible with V7 of the Unix operating system. MINIX comes with a C compiler that is K&R compatible (no floating point support, or source code), and a shell that is functionally identical to the Bourne shell. Over 175 Unix utilities are included as well as 200 library procedures. The distribution includes full source code (in C) for the operating system, utilities and libraries.

A book by Andrew S. Tanenbaum describes operating systems in general and MINIX in particular. The descriptions and source listings in the book are for the early version of MINIX, but are still useful, if not required, for understanding MINIX's internals. The book is:

Title:	Operating Syetems: Design and Implementation
Author:	Andrew S. Tanenbaum
Publisher:	Prentice-Hall
ISBN:	0-13-637406-9

MINIX is sold by Prentice-Hall. Some software stores stock MINIX, or it can be ordsred directly from Prentice-Hall.

Prentice-Hall sells MINIX for only a few machines. These are:

IBM (5 1/4"),	ISBN 0-13-585076-2
IBM (3 1/2"),	ISBN 0-13-585068-1
Amiga	ISBN 0-13-585043-6
Atari	ISBN 0-13-585035-5
Macintosh	ISBN 0-13-585050-9

All versions cost $169.00 plus tax and shipping. Included in this price is executable binaries, a detailed manual, complete source code (on diskettes), and a typeset; cross-referenced listing of the operating system. Prentice-Hall can be contacted at:

Microservice Customer Service
Simon and Schuster
200 Old Tappan Road
Old Tappan, NJ 07675
Phone Orders: (800) 624-0023 or (201) 767-5969

Even though source code is included, MINIX is not public domain, Prentice-Hall has copyrighted it. However, they grant permission for universities to copy the software under certain circumstances. Also, MINIX owners are allowed to modify the operating system and distribute 'diff' listings. The shrink wrap license allows the owner to make 2 backup copies.

If you're reading thie there ie a good possibility you do not own one of the computers for which Prentice-Hall is distributing MINIX. However, all ie not lost. MINIX has been ported to several other computers; such as the PT68-K2 via modifications called 'diff' listings. The hardest part of getting MINIX for one of these "other" computers is determining if there is a "bootable" 'diff' listing available.

As mentioned earlier, MINIX is copyrighted. It is not legal to distribute copies of the source code, even if modified for another computer. Prentice-Hall does allow the dietribution of these 'diff' listings; which are "different" from, and modify, the available commercial versions. To get one of these 'diff' versions of MINIX working, locate the 'diff' listing for your machine, purchase the proper commercial version for the 'diff' listing you need to use, apply the 'diff's to the commercial version which rebuilds the operating system, and load it to a disk bootable on your computer. (Be sure to purchase the correct commercial version. The Atari 'diff's will only work with Atari commercial version, and the compiled result will not work on any other computer.

There is no official source for the MINIX diff's. Most of the activity occurs on or through the USENET discussion group, comp.os.minix. In addition to questions and answers, numerous programs and 'diff' listings are posted regularly. This group is also carried as a BITNET mailing list. The activity from this discussion group is archived at numerous sights, so past postings can be downloaded via FTP or LISTSERV.

For those without access to USENET or BITNET there is a BBS that carries all traffic from comp.os.minix: The Mars Hotel BBS 301/277-9408; 300, 1200 and 2400 baud, 8, n, 1. No registration. Everyone gets 60 minutes a day. No upload/download ratios.

PT68-K ownsrs can obtain everything they need to get the Atari version running on a PT68-K from a bulletin board operated by Michael Everson, 817/488-8398; 300, 1200 and 2400 baud; 8, n, 1. No registration, no time limit.

Once the diffs you need are obtained, you must create a version of the operating system that will boot on your machine. This is greatly simplified if a "boot-disk" for your target system is available. A boot-disk is a listing of the MINIX kernel that will run on the target computer. Once running, the source code can be unpacked and the 'diff's applied.

Without a "boot-disk", MINIX must be running on the computer from which the port was made. The 'diff's can then be applied to the sources and the operating system recompiled. The executable modules must be transferred to the target computer and saved on disk. You must then create a disk that will allow MINIX to boot. This format will vary from machine to machine, but if you have network access I'm sure help can obtained there.

If you are porting to a 680xx (68020, etc...) processor, the vast majority of the code will not have to be modified. Most of the work will be in rewriting portions that handle the hardware (Interrupts, Clock Timer, Keyboard, Display, Disk Drives, etc.). After recompiling the code, the process will be identical to getting a version with 'diff' listings working (transfer to the target computer, and create a boot disk).

I understand that porting MINIX to one of the more powerful 680xx processors requires quite a bit more work due to a difference in interrupt/trap handling.

HARDWARE REQUIREMENTS FOR MINIX

Many people run MINIX on an IBM XT with 512K and two floppies. This is what I would call the absolute minimum. You could get MINIX running and do some work with it, but you would be severely restricted. I am not sure if you would be able to recompile the operating system with this configuration, but even if you could the disk swapping would be horrendous. Of course the larger the system the more you can do. I would recommend the following for a practical lower limit:

1 megabyte ram, 1 floppy drive, 1 20Mb hard drive and a text only monitor.

Currently a text mode monitor is the only option available. Some graphics routines have been developed (there is even work on bringing over X-Windows), but text mode is all that is supplied in the official releases.

Locating and obtaining MINIX for a non-supported system is really the biggest problem. Once you have it running, it works very well. The biggest complaint I've heard about the 68000 version of MINIX concerns its method of memory management. Because it does not use a memory management unit, MINIX keeps multiple processes running by copying them around in memory. Under most circumstances this is not as big a problem as it might seem, but there are instances where this creates a severe degradation in performance (most notably this occurs while running Kermit).

Is it worth it? Personally, I'd answer with an unqualified YES! Although $169.00 might seem a little steep for an operating system, when you consider all that is included it is really a bargain. Not only do you get the operating system but source code, a C compiler, and several editors! The fact that MINIX is so much like Unix allows many public-domain Unix

PAT
Full Screen Editor
for
Programs and Text

Features:
Paragraph Formatting, Centering
Search and Replace
Block: Move, Copy, Delete
Write Block to File
Insert at Cursor from File
Word Wrap
Auto Indent
Long Lines (up to 126 characters)
(screen shifts left to right)

$50.00 Specify- PT68K-2 or -4 SK*DOS REX
Video Monitor & IBM Keyboard
or
Serial Terminal Type
Desired Disk Format

Personal Checks Accepted
No Credit Cards Please

R.W. Anderson
3540 Sturbridge Ct.
Ann Arbor, MI 48105

source programs to be compiled without modification.

The package is an excellent introduction to advanced operating system concepts. With MINIX you get hands-on experience with OS design. Even if you never have a need to work with the source code, MINIX will provide you with an understanding of many of the system administration issues involved in the operation of a large scale OS.

Finally, asids from the educational value, MINIX is a very powerful operating system. Although it is not as visually appealing as something like Windows or Mac-OS, operationally it is more powerful. MINIX is very much a programmers environment. If you hope to buy applications to run with MINIX, you will be disappointed. If you want to write your own applications you will find MINIX an ideal OS to work in.

If you've read this far, you realize that getting MINIX to run on a computer not supported by Prentice-Hall is not a trivial exercise. However, with persistence and ingenuity you too can do it. My own experience was not smooth (this seema to be the norm). I do, however, have a floppy disk version running and am working to get a hard disk up. The tale of my journey to booting MINIX will be the subject of a future article. In the meantime, if you havs any questions feel free to write to me c/o "68xxx Machines". If you have an address on BITNET, Internet, or CompuServe please include it. JF

Inexpensive Flex Floppy Drives

by Alen Gordon, MD

This articls shows how to convert 5 1/4" and 3 1/2" inexpensive Teac brand 1.2Mb floppy drives intended for IBM system use, for use in Flex based systems. Drives used this way will format floppies which are compatible with floppies formatted on older Flex systems. I have never used any brands other than Teac, but this information may apply to other brands as wsll.

As the capacity of floppy drives became larger one step was double-density, and double-sided. Early drives had 40 tracks each side and were called Quad density. The nsxt increase was to 80 tracks on each side, and was called Octal Density. Octal density drives are formatted to 1Mb total capacity. Of this, the user receives 2844 sectors of 256 bytes per sector; or 728,064 bytes. IBM calls this format standard, 720K.

The main difference between the IBM 1.2Mb drive and other drives is the rotation rate. The 1.2Mb turns at 360 RPM, all other 5 1/4 inch drives turn at 300 RPM. All the 3 1/2 inch drives turn at 300 RPM as well. Bench testing floppy drives will demonstrate this by measuring the time between two index pulses on pin 8 of a drive. The 'all other' drives have

SK*DOS ® /68K Upgrade Offer

Why make do with a me-too DOS when you can upgrade to the new, faster SK*DOS and get all this at a special price: Simple-to-use DOS with full documentation and on-line help • Multiple directories • Multiple I/O devices and drives • User-installable device drivers • Keyboard type-ahead • Print-screen • RAM disk • Disk cache • 5¼" and 3½" floppy disk support, normal or high density • Easy hard drive partitioning • I/O redirection to drives or I/O • Time/date stamping of files • file or disk write-protect (even hard disk) • Batch files • Monochrome or color video board support • Read and write MS-DOS disk files • Emulator for running a wide variety of 6809 programs • Powerful utilities such as copy-by-date, undelete, show differences between files, prompted delete, text file browse etc. – all included • Basic interpreter included • Fast assembler included • Editor included • User support via newsletter, BBS and users' group • Available third-party software includes C and other compilers, full Basic, screen editors, disassemblers, cross-assemblers, text formatter, ROM-based debugger, modem communications program, screen-based shell and hard disk manager program, and more.

SK*DOS is now priced at $70, but here is a special upgrade offer for new users: trade in any old 68K DOS and get SK*DOS for just $50.

(Note to current SK*DOS owners: as always, your upgrade price remains at $5.)

Star-K Software Systems Corporation
P. O. Box 209
Mt. Kisco, NY 10549
Phone/Fax: (914) 241-0287 • BBS: (914) 241-3307

a period of 60/300 which equal 200ms between pulses. The 1.2Mb drives will show 60/360, or 170ms between pulses.

This attempt by IBM to increase the capacity of the 5 1/4 inch drive in this way was one of the worst ever made. The 3 1/2 inch drive increases its capacity from 720Mb to 1.44Mb by leaving the rotation rate at 300 RPM, while using better media and increasing the data transfer rate by 100%.

For some reason the 5 1/4 inch drive went from 720K to 1.2Mb (An increase of 67%) by increasing the rotation rate by 20% and increasing the data transfer rate to account for the balance. Fortunately for us the newer 1.2Mb drives will still handle lower data transfer rates. All that is needed is to make the drives turn at 300 RPM. (Of course, the 1.2Mb drives are widely available and cheap.)

Teac Octal density drives had names like FD-55SR. Current 1.2Mb drives have names like FD-55GFR. As far as I know, the 'F' means the drive turns at 300 RPM. 'G' means the drive turns at 360 RPM; which makes it IBM compatible. 'GF' means the drive is convertible to either.

Many floppy drives are manufactured to be configurable to several different formats. To make it easy to do, the drive's PCB (printed circuit board) is prewired with each possibility, but are left unpowered. To invoke a given possibility a little device called a shorting block is slipped over two pins. These blocks complete the connection between the pins and thereby complete the circuit.

Shorting blocks come in different sizes; be sure to use the proper size. They can be obtained at most electronic stores.

The pins the shorting blocks are slipped over project out of the PCB. They have identifying letters and numbers printed next to them on the PCB.

To make a FD-55GFR drive turn at 300 RPM slip a shorting block over the two pins labeled 'I' and another over those labeled 'LG'; (Leave the existing blocks in place). That is all there is to it! You now have an unlimited supply of brand-new 5 1/4 inch drives for less then $80. It will even read and write your cheapest older single sided floppies at 720K.

Adding a shorting block to the Teac 3 1/2 inch drives at 'HHI' disables the drive's ability to tell the difference between 720K and 1.44Mb floppies. Any floppy looks like a 1.44Mb format when 'HHI' is shorted. When done, this seems to put a 1000 Ohm resistor across the switch which senses if the floppy is 720K or 1.44Mb.

With 'HHI' shorted you can put any 720K floppy in the drive and format it to 1.44Mb. The disadvantage is, if you put this same floppy in other drives, it will look to be 720K, and be unreadable. Of course 1.44Mb floppies will read and write correctly, and the data can be transferred to other drives.

Pin spacing on Teac 3 1/2 inch PCB drives is not 0.1 inch. Forcing regular 0.1 inch shorting blocks will ruin the drive's pins. If you have any questions or comments, please write, inclosing a SASE, to 160 NW 176 St, Miami FL 33169

Micronics Research
RBASIC
Enhanced BASIC Interpreter
with built-in Line-Editor
for 6809 or 68000 FLEX/SK*DOS

US$100 + $5 Shipping/Handling
continental USA and Canada
($10 S/H elsewhere)

**** Please specify Disk Size and Format ****
(e.g., 5-inch 80-track)

Sorry! No credit cards!
Checks may take 2 weeks to clear

Please make Bank Draft/Money Order payable to:

R.Jones, 33383 Lynn Avenue, Abbotsford,
B.C., CANADA V2S 1E2 (604)854-6814

SYSTEM IV

SYSTEM IV - The 68000 Computer already serving customers here and abroad!

- Expandable using readily available low-cost cards
- Flexible enough to tailor to your requirements
- Versatile enough to use OS-9/68000, OS-9/6809, MS-DOS and other Operating Systems
- Powerful - outperforms other machines in its' price class

THE BEST OF ALL WORLDS

1 - A Multi-User Multi-Tasking System
2 - Optional Plug-in Board for MS-DOS
3 - Optional Emulator/Interpreter for OS-9/68000 Software
4 - Will support Other Operating Systems

Just call, FAX, or drop us a line. We will respond promptly with complete pricing and specifications.

DELMAR CO - serving its' customers since 1975.

Terminal Systems	from $999.00	Console Systems	from $1,149.00

Assembled boards and kits available

delmar co

Middletown Shopping Center - PO Box 78 - Middletown, DE 19709
302-378-2555 FAX 302-278-2556

OS9/68000 SOFTWARE

M6809 - 6809 Emulator/Interpretor **$ 99.95**
M6809 emulates the user mode OS9 Level II operating system and interprets OS9-6809 binary object modules under OS9-68000.

QUICK ED - Screen Editor and Text Formatter **$275.00**
A high quality documentation tool and program editor ideally suited to laser printer users. Uses function and cursor keys on any terminal, configurable per user. Micro-justifies mixed proportional text. Automatic table of contents generation and user-definable macros and commands. Handles an unlimited number of fonts. Drives any printer. Ideal for multi-user systems. Available on a 30-day try before-you-buy basis.

FLEXELINT V4.00 - The C Source Code Checker **$495.00**
Flexelint finds quirks, idiosyncracies, glitches and bugs in C programs. 60 options control checking by symbol name or error number. Checks included intermodule inconsistencies, definition and usage of variables, structures, unions and arrays, indentation, case fall-through, type conversion, printf and scanf format string inconsistencies, and suspicious semi-colons. A must for all serious C programmers.

DISASM_OS9 - OS-9/68K Disassembler **$250.00**
This high-speed, three-pass 68000 disassembler can also handle the 68010 and 68020. It intelligently decodes module headers and produces symbol information that can be repeatedly edited and passed through the disassembler allowing iterative disassembly. The system libraries are read to supply symbols.

IMP - Intelligent Make Program **$250.00**
IMP does everything you wished Microware's Make would do, and a great deal more. It is well-behaved, consistent, and extremely flexible. It has a built-in C-like preprocessor and has comprehensive debugging facilities. Rules can be user-defined, and make files for jobs other than assembly-language or C compilation are easily constructed.

WINDOWS - C Source Code Windowing Library **$250.00**
This C source code library package supports multiple overlapping windows displayed on one character-based terminal screen. It supports window headers and footers, and pop-up windows. Windows may be moved, panned, written to while *off*-screen, etc.

PROFILE - User State Program Profiler **$270.00**
Designed to profile user-state programs. Profile effectively samples a traced execution building statisical information as it goes. It reads symbol table modules to give a function-by-function account of the time spent during execution. The user may "zoom-in" on a function to find a smaller range of addresses where time is being spent.

PC9 - MS-DOS to OS-9 Windowing System **$350.00**
PC9 allows an MS-DOS computer to be used as a terminal to multiple processes on a remote OS-9 system linked by a single serial cable. Each OS-9 process is displayed through a resizable, moveable window on the PC screen. Terminal emulation facilities support uMACS and other screen editors and provide a programmable PC keyboard. Access to PC disk drives is also avilable through the OS-9 unified I/O system, giving disk capability to ROM based OS-9 systems.

delmar co
Middletown Shopping Center - PO Box 78 - Middletown, DE 19709
302-378-2555 FAX 302-378-2556

Why Your Disk Drive Won't Work

by Ed Gresick

In the beginning of June I received the following letter from Bill Hughes of Valley Falls, Kansas. ED.

Editor:

I would be very interested in an article about sharing software among the various machines running OS-9 (or SK*DOS, REX, etc.) I have described my experience below; but I'm sure some of your regular contributors will have wider experience.

I have a PT68K-4 with a 3.5" and a 5.25" floppy drives. Originally both drives were running on the Western Digital 1772 floppy disk drive controller. The default OS-9 format for this machine is 10 sectors/single side on track 0; and 16 sectors per track/double sided for all other tracks (OS-9 "standard").

The problem began when I ordered software that was advertised for the CoCo. It was to run under OS-9, so I assumed there wouldn't be any trouble getting it onto my machine. The following points are notable:

1) The CoCo and the PT68K use different "standard" disk formats. I now know how to change the description in the IO/d0.a file, compile a new descriptor, and generate the proper boot file. However, getting this information from square one was not easy. It became a real learning experience for a beginner like myself.

2) The 1772 chip on the HD (high density) 5.25" doesn't seem to like CoCo disks, regardless of format. I later installed the newer 37C65 chip. It seems to handle anything (so far), so now I can share 5.25" disks (provided we agree on which format to use.)

3) The 3.5" drive on the 1772 controller reads/writes/formats (in PT68 standard format) disks that are readable/writable by CoCo.

4) I couldn't get the 1772 controller to give reliable results on either drive using CoCo format. 18 sectors per track may be too much for it to figure out.

5) Obviously, always use DD (double-density) diskettes: my 5.25" could handle HD diskettes without a problem, but I guess CoCo's are strictly DD drives.

I have been working with Bob van der Poel on this CoCo<->PT mess. He has also had experience with transfers between CoCos and MM-1s. He included his format scheme when he sent the software; which was very helpful, since it got me beyond the easy solutions pretty quickly!

In summary I would caution PT68K users to be careful about mixing and matching drives and controllers if they want to exchange disks with non-PT machines. Also, it might be a good idea for software companies to indicate what format they use (or, heaven forbid, adopt a STANDARD!).

This looked pretty serious to me so I forwarded it to Ed Gresick of the DELMAR CO, the distributors of the PT computers. The following is his very informative reply. ED.

Dear Jim:

I've reviewed Mr. Hughes' letter. He is correct with respect to mixing drives and controllers. But this is not a PT problem; rather it's a problem on all computers. On the various forums and BBSs, people ask how they can use their new drive (usually already purchased) with such-and-such system. All too often, the mix won't work.

Experienced OSK software companies will ask the customer what disk format the customer prefers.

Attempts to mix a 1772 controller chip with HD (high-density) 5 1/4" drives is an invitation to disaster. They are not compatible. The 1772 controller chip has a data transfer rate of 250KBits/sec. A HD 5 1/4" drive will transfer data at 300K Bits/sec because it is rotating at 360 rpm unless it is instructed to change to 300 rpm. *(Don't miss Alen Gordon's article in this issue. ED.)* Not all HD 5 1/4" drives are able to change speed although I think most may be modified to do so. The 1772 is unable to support this kind of switching. Peripheral Technology has solved the HD 5 1/4" speed problem by specifying a 300Kbit/sec transfer rate but this works only with the 37C65 controller chip. This solution avoids changing the drive speed.

HD 3 1/2" drives detect the type of disk and adjust their speed accordingly so they might work with a 1772. The 37C65 controller chip is capable of handling different transfer rates accounting for the success Mr. Hughes achieved with this combination.

Through and including version 2.3 of OS9/68000, Peripheral Technology used standard Microware format. This format is 10 sectors per track, track 0, side 0 and 16 sectors per track for the remaining tracks (including track 0 side 1). With version 2.4, Microware changed their standard format to 'universal' format. Peripheral Technology has followed suit and is using the 'universal' format as their standard format for 720K drives. One other common format in use is the MIZAR format. This format is 16 sectors for all tracks and is popular in industry and business.

The SYSTEM IV, manufactured by Peripheral Technology for DELMAR CO, uses 28 sectors per track with a 5 1/4" drive and 34 sectors per track with a 3 1/2" drive. HD drives are standard. Should a customer not want HD drives, we do provide 720K drives and follow the Microware standard with the universal format for the distributions disks. Alternate formats are available upon request. The HD configuration was selected because Western Digital has discontinued the 1772 controller chip, 720K drives are becoming difficult to obtain and the rest of the computer industry has shifted to HD 3 1/2" drives and HD 5 1/4" drives. We provide descriptors to handle the various 720K formats for backward compatibility. As long as they are available, the 1772 controller chip can be provided as an option.

The CoCo format (18 sectors per track) is not a Microware format nor do they recognize it. (See the 'rbfdesc.a' file in the IO directory or the 'systype.d' file in the DEFS directory.) This format was devised by Tandy to conform with their RSDOS format. While Microware did write the original OS9 6809 operating system, Tandy's license permitted them to make changes and additions which they did. All support was the responsibility of Tandy.

Mr. Hughes stated he ordered software '... that was advertised for the CoCo, but was OS9, ...'. CoCo OS9 software is not compatible with OS9/68000. The instruction set used in the respective processors is entirely different. There is one exception, CoCo Basic09 programs will usually run under OS9/68000 Basic with slight or no modifications so long as graphics are not required.

I suggest Mr. Hughes upgrade to version 2.4 of OS9/68000 if he hasn't done so already. A complete set of descriptors for HD and 80 track DD disks for universal, the old Microware standard, MIZAR and Tandy Color Computer are provided for 3 1/2" and 5 1/4" HD drives. The upgrade is available from PT or we can provide it. Starting with version 2.3, Microware included a neat utility 'moded'. This utility allows editing the descriptor object code in the CMDS/BOOTOBJS directory without having to assemble them each and every time.

There are two caveats Mr. Hughes should be aware of. If DD, 720K disks are to be exchanged between a HD and a DD drive, format the disks on the DD drive. Formatting on the HD drive may yield disks that are unreadable by a DD drive. This is not an OS9/68000 problem; rather it is a hardware problem having to do with the magnetic coating on the disks and the head drive currents. And, do not use HD disks in a DD drive - the results are unpredictable.

I hope I've responded satisfactorily. If not, give me a yell!

Best regards,

Edward Gresick, Pres.
DELMAR CO
PO Box 78
Middletown Shopping Center
Middletown, DE 19709
Phone 302/378-2555
FAX 302/378-2556

Keep tuned, if all goes well next month we'll have a complete answer to Bill's first sentence. ED.

ADVERTISER'S INDEX

The Tech Corner

by J. Scott Kastner

When we left off last time, we were developing a mini graphics library to help us explore the techniques and problems involved with modern computer graphics. The example code for the library was developed on a Peripheral Technology OSK system with Super VGA. I can't stress highly enough to substitute in the code for YOUR system if you have something different. The resolution that we will be working with most often is the VGA 640 X 480 mode in 16 colors, just pick a reasonably high resolution if yours does not support that mode.

The method for calling the Peripheral Technology graphics driver is through the I$SetStt system call; the parameters are all held in CPU registers. Here is a brief description of the two we are using:

```
All calls.
     d0.w - path number
     d1.w - function code

SS_SetMode - Set display mode.
     d2.b - mode value

SS_WritePix - Set a pixel.
     d2.b - color
     d3.w - x position
     d4.w - y position
```

Last time, we coded the Graphics() routine in assembly using the #asm #endasm options of the Microware C compiler. The routine to set a pixel is a little more complicated; we will use a mix of C and assembly. Add this routine to the 'gfx.c' file started last time:

Listing #2a for gfx.c

```
/*-------------------------------
Routine to clip and plot a pixel.
-----------------------------*/

int Plot(x,y,c) register int x,y,c; {
/*Return FALSE if off screen.*/
  if ((x<0)  (y<0)  (x>639)
       (y>479)) return 0;
  {}

/*Plot and TRUE if on screen.*/ #asm
* Save registers for later.
```

```
  movem.l d0-d4,-(sp) * Set path and
function call.
  moveq.l #1,d0       StdOut
  move.w   #$85,d1        SS_WritePix *
Copy args to proper regs.
  move.b  d6,d2       copy color
  move.w  d4,d3       copy x pos
  move.w  d5,d4       copy y pos * Make
sys call.
  OS9   I$SetStt * Recall regs.
  movem.l (sp)+,d0-d4 #endasm

  return 1; }
```

This routine demonstrates the use of register variables. When we enter the Plot() routine, the three arguments are contained in d0, d1, and one on the stack as described last time and in section 3-1 of the Microware C compiler manual. The 'register' prefix caused the compiler to generate code copying the parameters into registers d4, d5, and d6. The values will be accessed from there instead of on the stack or something, like the compiler would normally code it. This is handy for us since we need to use them directly with the inline code. The inline code copies the values to the appropriate registers and makes the sys call to the driver. On return from Plot(), there will be a value in register d0. This value will be a 0 (FALSE) or 1 (TRUE) to indicate whether a point was actually plotted or not due to the edge of screen clipping (the if statement). We could have done this as in listing #2b, but it is easier and sometimes clearer to use the C code when one can.

Listing #2b for gfx.c

```
/*-------------------------------
Routine to clip and plot a pixel.
------------------------------*/

#asm Plot: * Is off in X?
  cmpi.w  #0,d0       x<0 ?
  bge.s   Plot_1
  clr.l   d0          rtrn FALSE
  rts Plot_1
  cmpi.w  #639,d0     x>639 ?
  ble.s   Plot_2
  clr.l   d0          rtrn FALSE
  rts * Is off in Y? Plot_2
  cmpi.w  #0,d1       y<0 ?
  bge.s   Plot_3
  clr.l   d0          rtrn FALSE
  rts Plot_3
  cmpi.w  #479,d0     y>479 ?
  ble.s   Plot_4
  clr.l   d0          rtrn FALSE
  rts  *  Save registers for later.
Plot_4
  movem.l d0-d4,-(sp) * Copy args to
proper regs.
  move.b  27(sp),d2 copy color
  move.w  d0,d3       copy x pos
  move.w  d1,d4       copy y pos * Set
path and function call.
  moveq   #1,d0       StdOut
  move.w   #$85,d1        SS_WritePix *
Make sys call.
  OS9   I$SetStt * Recall regs.
  movem.l (sp)+,d0-d4 * Return TRUE.
  moveq   #1,d0
  rts #endasm
```

The assembly version is a functional equivalent to #2a. Since it was all assembly, extra code was needed to get the arguments that were passed, and to generate a return value. The color value was on the stack so we had to fetch it with the move.b 27(sp),d2 instruction. How did I get the value 27 as the offset for this? Well, we pushed d0 through d4 on the stack ourselves. The CPU put a return address on the stack from the BSR or JSR instruction that called this routine. This is a total of 6 long word values, 6*4 = 24. The color value is the first argument on the stack after this, but it was converted to a long before being placed on the stack, the byte we want is three away from the start, so we now have 6*4 + 3 = 27. Now it should be clear why #2a is a little more clear and certainly less work.

The last routine to be added is the one to return us to the text mode. Just add listing #3 to your 'gfx.c' file.

Listing #3 for gfx.c

```
/*-------------------------------
Routine to return to text mode.
------------------------------*/

#define Text_Mode_80 0x03

Text() {
  Graphics(Text_Mode_80); }
```

This routine just makes a call to the graphics mode routine we created last time. The 'Text_Mode_80' is just a mode value for the 80 column text screen. The 0x03 is a way of writing hexedecimal notation in C.

At this point, we have covered quite a bit of material, so I'll break here and let you digest it some more. As far as chapter 3 of the Microware C manual, READ IT, LIVE IT, LEARN IT, it's your key to really getting the most out of a superlative C compiler. Next time we will create the header file for the library, compile it, and do a test drive. Until then, I'll leave you with a mystery. Why are the two empty braces {} in listing #2a necessary for the code to work right? I'll give the answer next month.

Beginner's Corner

by Ron Anderson

Last time (Apr '91) we got as far as a filter program under SK*DOS, and did quite a few variations on a theme. Let's talk a little this time about simple utility programs. For example, suppose you have an Epson printer attached to your system and want to set it to the "Emphasized" mode. The Epson manual indicates you have to send it an "escape sequence" consisting of the ESCape character followed by a capital "E". The following program will do that just fine:

```
 * PROGRAM TO SET EPSON TO EMPHASIZED
MODE
 *
 PUTCH EQU $A033
 VPOINT EQU $A000
 WARMST EQU $A01E
 DEVOUT EQU 3275
 *
 START DC VPOINT SET A6 TO POINT AT
SK*DOS VARIABLES
  MOVE.B #2,DEVOUT(A6) SELECT DEVICE
2 FOR OUTPUT
  MOVE.B #$1B,D4 $1B IS THE CODE FOR
THE ESCAPE CHARACTER
  DC PUTCH OUTPUT IT
  MOVE.B #$45,D4 $45 IS THE CODE FOR
E
  DC PUTCH
  MOVE.B  #0,DEVOUT(A6)  SET  OUTPUT
DEVICE BACK TO TERMINAL
  DC WARMST RETURN TO SK*DOS
  END START
```

The only possible point of confusion here is the DEVOUT memory location. If you have a printer, your STARTUP.BAT file probably has a call to the DEVICE utility of SK*DOS. You have probably installed a printer driver as device 2. If your printer is installed as a device other than 2, the line below START in the above program must be changed so you move the correct device number into the DEVOUT location. DEVOUT is a variable which is part of SK*DOS. It contains the number of the device used by PUTCH to output a character.

SK*DOS also has a variable named DEVIN which serves the same function for input. Assuming you have a modem installed on a serial port as device 4, you could put a 4 in DEVIN (MOVE.B #4,DEVIN(A6)) and receive input from the modem just as though it were your terminal.

Of course you would have had to define DEVIN with an EQU statement as we did for DEVOUT at the beginning of this program. To do something like this in a higher level language would take quite a few more bytes of code and would probably involve a little assembler code as well. Anyway, since the location of DEVOUT is relative to the start of the SK*DOS variables; which should be determined by a call to VPOINT.

It is generally easier to "massage" the operating system directly in assembler code. There are all sorts of simple utility programs you can write for yourself. For example, when you go away from your terminal for some time, it is easier on the screen if it is blank or almost so. I wrote a quick program called CL to handle that for me. On one of the 68000 systems I use, we have a serial terminal. It is an "ANSI" standard terminal which emmulates a VT-100. The code sequence to clear the screen and put the cursor on the first line is rather complex but it will serve to illustrate how easy it is to do.

```
 * PROGRAM TO CLEAR SCREEN AND HOME
CURSOR ANSI TERMINAL
 *
 PUTCH EQU $A033
 WARMST EQU $A01E
  ORG $0000
 START MOVE.B #$1B,D4
  DC PUTCH
  MOVE.B #'[',D4
  DC PUTCH
  MOVE.B #'2',D4
  DC PUTCH
  MOVE.B #'J',D4
  DC PUTCH
  MOVE.B #$1B,D4
  DC PUTCH
  MOVE.B #'[',D4
  DC PUTCH
  MOVE.B #'1',D4
  DC PUTCH
  MOVE.B #';',D4
  DC PUTCH
  MOVE.B #'1',D4
  DC PUTCH
  MOVE.B #'H',D4
  DC PUTCH
  DC WARMST
  END START
```

A simpler program clears the screen of most terminals such as the Televideo or Wyse non ANSI versions.

```
 * PROGRAM TO CLEAR SCREEN AND HOME
CURSOR
 *
 PUTCH EQU $A033
 WARMST EQU $A01E
  ORG $0000
 START MOVE.B #$1B,D4
  DC PUTCH
  MOVE.B #'*',D4
  DC PUTCH
  DC WARMST
  END START
```

You might look at these programs, particularly the first, and say we've done it very inefficiently. We could define a string of bytes terminated with a 4 and call the SK*DOS system routine PNSTRN, which outputs a string without a leading CRLF. Just for fun, let's do it that way and get a feel for how much we could save in program code and bytes.

```
 * PROGRAM TO CLEAR SCREEN AND HOME
CURSOR ANSI TERMINAL
 *
 PUTCH EQU $A033
 PNSTRN EQU $A036
 WARMST EQU $A01E
  ORG $0000
 START LEA STRING(PC),A4
  DC PNSTRN
  DC WARMST
 STRING FCC $1B,"[2J",$1b,"[1;1H",4
  END START
 ,p
```

At this level of program, it really doesn't matter much. The whole program is less than one sector of disk storage, and a whole sector would be used for any program from 1 to about 245 bytes. When you run this CL program you won't be doing anything else, so memory usage is not important either.

As a matter of interest the first "long way" program is 62 bytes and the one directly above, which works identically is 19 bytes. I should mention I have been using the long one for a year or more, primarily because it was a simple extension of the non-ascii clear program that had been done first. Looking at it again today brought to mind there was a better way.

I find with programming there seems to be no end of looking at a program and making it better and shorter or faster. The improvements come more slowly and make less change in the performance or size of the program after you have worked on it for a long time, but you never reach the point where improvements cannot still be made. (Well, of course, for a very simple two line program, you do reach that point, but for any non-trivial program, it seems to be a never ending process).

Now looking at the program's content, obviously for the non-Ascii terminal, ESC * is the sequence which clears the screen and homes the cursor. The ANSI version requires: ESC [2J to clear the screen and ESC [1;1H to put the cursor at row 1 and column 1 of the screen. The Hercules board video version for the PT68K-2, which probably most of you have, uses the code $1A (control Z). You only need to send it one character, as below:

```
 * PROGRAM TO CLEAR SCREEN AND HOME
```

PT68K2/4 Programs for REXDOS & SK*DOS

EDDI	A screen editor and formatter	$50.00
SPELLB	A 160,000-word spelling checker	$50.00
ASMK	A native code assembler	$25.00
SUBCAT	A sub-directory manager	$25.00
KRACKER	A disassembler program	$25.00
NAMES	A name and address manager	$25.00

Include operating system, disk format, terminal type and telephone number with order. Personal checks accepted. No charge cards.

PALM BEACH SOFTWARE
Route 1 Box 119H
Oxford, FL 32684
904/748-5074

```
CURSOR
*  FOR  PT68K-2  USING  MONOGRAPHICS
BOARD
*
 PUTCH EQU $A033 WARMST EQU $A01E
 ORG $0000 START MOVE.B #$1A,D4
 DC PUTCH
 DC WARMST
 END START
```

Well, there you see some of the usefulness of Assembler programming, particularly with regard to massaging the operating system to make it do what you want it to do. We've looked at filter programs, found out how to open and close disk files, and talked to the printer and terminal with simple utility programs. Next time we'll look at higher level languages and see why we might not find them appropriate for all of our programs.

The Eccentric Programmer

by Randy Krippner

A famous computer scientist once said the comment that using BASIC causes brain damage. I find that comment absurc. I am a BASIC programmer and I can assure you that my brain isn't damaged in the slightest. (The rumor that I entered a Madonna look-alike contest and took second place is a vicious rumor. I took first.) Just because someone chooses to program in BASIC doesn't mean that his or her RAM bank is a few chips short of a full meg.

One thing that **can** make an otherwise normal person feel a sudden urge to go live with the squirrels is trying to learn a new programming language. Once you have become thoroughly familiar a language, it can be difficult to adapt to a new one. This is true even when changing from one veraion of a language to another. In fact, moving from one dialect of a language to another can sometimes be more difficult than trying to learn an entirely different language. This is especially true when trying to move from a 'standard' BASIC like Extended Color BASIC to a modern version like Basic09.

If you've decided to take the plunge and switch from ECB to Basic09, you're in for a surprise. BASIC ain't BASIC any more, folks. Take a look at the two simple code fragments below:

```
10 REM JUNKLOOP.BAS
20 FOR CT = 1 TO 10
30 PRINT "Hello"
40 NEXT CT
50 END

PROCEDURE JunkLoop
DIM count:INTEGER
FOR count = 1 TO 10
  PRINT "Hello"
NEXT count
END
```

These two programs, the first in ECB, the second in Basic09, do exactly the same thing, but there are some significant differences in *how* they do it.

The most obvious difference ie that the Basic09 program doesn't have line numbers. Line numbers are optional in Basic09. They are required only when a line is referenced by a GOTO or GOSUB. Otherwise they can be used or not, depending on the programmer's preference.

The PROCEDURE line of the Basic09 listing is also something new. Every Basic09 program starts with the word PROCEDURE, followed by the name of the program or module.

The PROCEDURE line ie inserted automatically by the Basic09 editor whenever you atart to write a new program. To start editing a new program, you would type:

```
e JunkLoop [ENTER]
```

The "e" tells Basic09 to go into the edit mode. The word after the "e" is the name of the module you want to edit. If the module does not exist, Basic09 will create it, giving it the name you typed. If a module of that name already exists, Basic09's editor will allow you to edit that module. If you do not enter a name after the "e" command, Basic09 will give the procedure a name of "program".

If ECB is the only language you've ever used, you may find the **DIM** statement in the second line. In ECB, DIM is used only to dimension an array, as in **DIM A(10)**. In Basic09, however, **DIM** serves two purposes. It is used to dimension arrays, just as in ECB, but it is also used to declare *variable types*.

In ECB, there are only two types of variables; numeric and string. Basic09 has ***five*** different variable types; integer, real, byte, string, and boolean.

In Basic09, *real* variables and *string* variables are similar to ECB's numeric and string variable types. **Real** variables are used to hold standard floating point numbers. **String** type variables hold, of course, strings. However, string variables under Basic09 are a bit different from what you may be used to.

Integer variables are numeric variables also, but they can only hold whole numbers in the range of -32768 to +32767. *Byte* variables are also numeric, but the values they can hold are limited to the range of 0 - 255.

Boolean variables are used exclusively for logical comparisons. They can hold only one of two values, **TRUE** or **FALSE**. Boolean variables are usually used as flags to test to see if specific conditions have been met.

And as if that isn't enough, Basic09 also supports user defined data types, something we'll get into a bit later.

Why all of these different variable types? It may seem that all of these different types would confuse things, but they actually help programmers in the long run.

These different data types can make a program much more efficient in terms of execution speed and memory use. Look at JunkLoop again. I've declared *count* to be an integer variable. Since it is an integer, it takes up only two bytes of memory, rather than the five bytes a **real** variable would have taken up.

Also, since it is an integer rather than a floating point value, mathematical operations are performed much more quickly. Performing manipulations on floating point numbers takes a great deal of time.

(Note: You have to make sure that the number being stored in the variable will not exceed the capacity of the variable's type.)

What happens if you do *not* declare a variable with a DIM statement before you use it? Well, unlike Pascal and C which require *all* variables to be declared before they can be used, Basic09 will allow you to create new variables on the fly, so to speak. Undeclared numeric variables default to the real type. Undeclared string variables default to a string length of 32 characters. (Unlike ECB which automatically adjusts the length of a string to match the string assigned to it, under Basic09 you have to set the maximum string length yourself. It may sound like an annoyance, but there are advantages. First, strings are not limited to 255 characters in length. A Basic09 string can be as long as you like (within memory limitations). The second advantage is that there is no need to endure the dreaded "garbage collection" delays for which ECB is so justifiably infamous. Because Basic09 strings are of a fixed length set by the programmer, the language doesn't have to pause to reorganize string memory.)

Another advantage of Basic09 is that you are no longer restricted to two character variable names. Variable names can be up to 32 characters long, all of them significant. You don't have to go crazy trying to figure out whether CT$ is the customer name, credit rating or something else. Under Basic09 you can give variables names that actually mean something useful, such as CustName or CreditRating.

Now let's talk about those four digit numbers that appear in front of every line in a Basic09 program. If you type in JunkLoop and then list it, you'll see something like this:

```
          PROCEDURE JunkLoop
    0000  DIM count:INTEGER
    0007  FOR count=1 TO 10
    0017      PRINT "Hello"
    0033  NEXT count
    003E  END
```

Newcomers to Basic09 often think these are line numbers. They aren't. They are actually hexadecimal ***addresses***. The number to the left of the line is the address of where that line of code starts in the computer's memory. (This is is not the actual memory location where the program is stored. Normally we don't know, and don't care, exactly where a program is stored in memory. OS9 and Basic09 take care of that for us. The addresses displayed in a Basic09 program are ***offsets*** from the beginning of the RAM area being used. For example, perhaps OS9 has given Basic09 a block of RAM starting at 0100 hex. The *offset* is added to the base address to produce the true address of where that particular item is stored. So the second line of JunkLoop is not really stored at 0007, but at 0107, or the base address plus the offset.)

Since exactly where a Basic09 program is located in memory is normally something we don't care about, why does Basic09 give us this list of addresses?

Well, the addresses can help you when you are debugging a program. Remember, we aren't working with line numbers, so it can sometimes be difficult to find a particular location in a program. If your program crashes, Basic09's built in debugger will give you an error message that includes an address, telling you where the problem turned up. You can compare that address with the list of addresses printed in the Basic09 listing to find where the error took place.

Next time we'll continue our move from ECB to Basic09. Questions can be sent to Randy Krippner, 1014 W. Hwy. 114, Lot 29. Hilbert, WI 54129. Please include an SASE.

"The 68xxx Machines"
The Chatham House Company
RD#1 Box 375
Wyoming, DE 19934 USA

WILMINGTON, DE
PM
JUL 11
1991
19850

- Address Correction Requested -

- First Class -